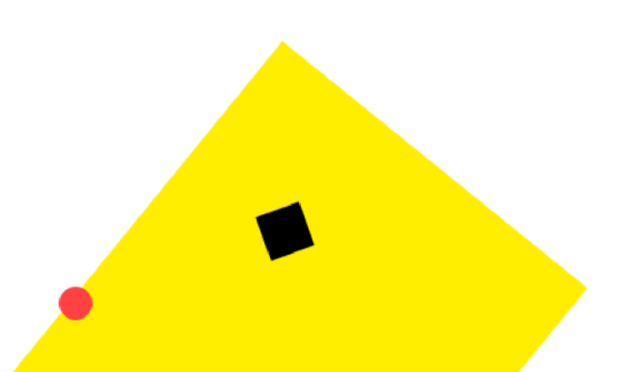
Как выстроить удобные процессы в работе с монорепозиторием

Андрей Кочеров (Яндекс Такси)





Команда

разрабатывает веб-приложения для таксопарков,
коллег в Яндекс, внешних сотрудников и исполнителей

выросла от **5 человек,** работающих над **одним проектом**
до **17** человек и **9** активных проектов
с общими **библиотеками** и **инструментами**
в едином репозитории



Проекты

один проект **разделился** на два
с общей бизнес-логикой

появились **новые** проекты
с общими компонентами, системой локализации,
инструментами разработки и деплоя

обобщение **в ходе разработки** новых проектов



Цель

относительно небольшой командой **эффективно** разрабатывать **несколько** проектов **одновременно**

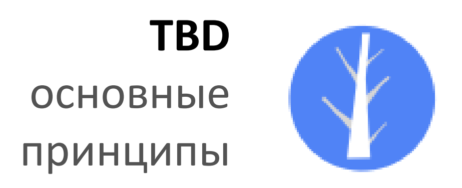


А для этого нужно

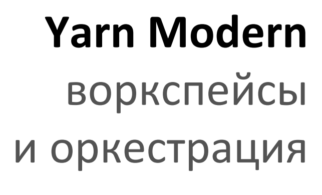
уметь **переключаться**
между проектами

легко **переиспользовать** решения
и **не замедляться**



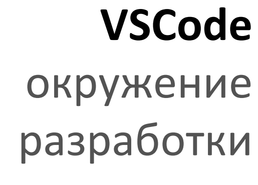


TBD
основные
принципы



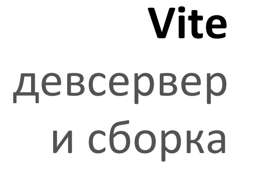
Yarn Modern
воркспейсы
и оркестрация





VSCode
окружение
разработки





Vite
девсервер
и сборка





Prettier
форматирование



TypeScript
IntelliSense & проверка типов

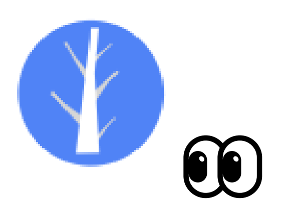


ESLint
ошибки логики



Jest & Vitest
тестирование













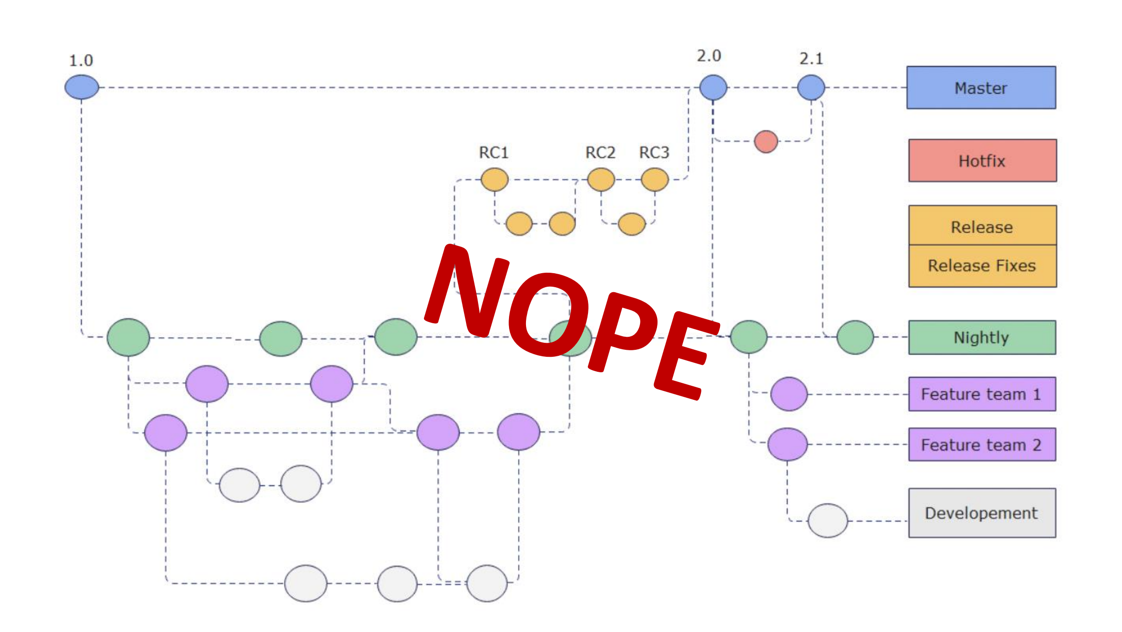

Trunk Based Development

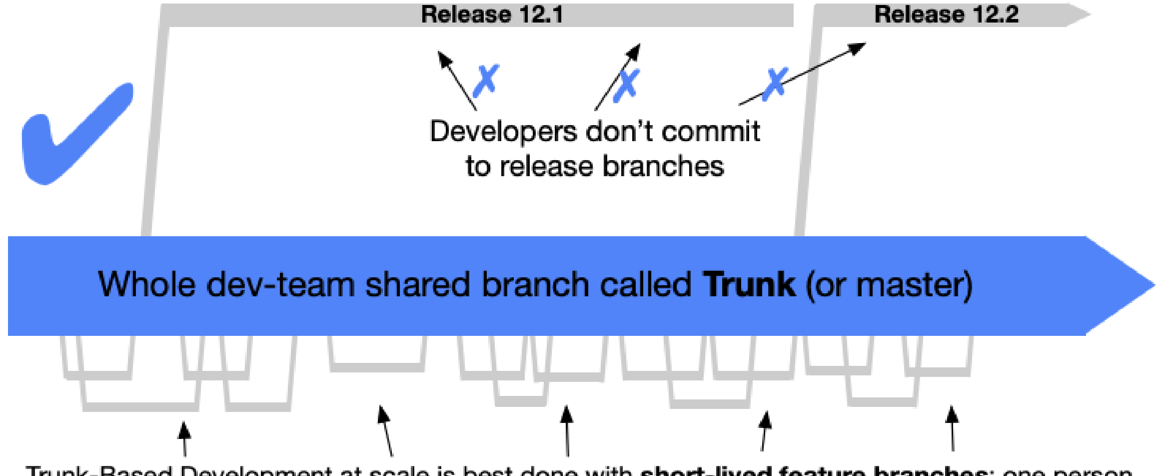
trunkbaseddevelopment.com



Trunk

общая для всех и **единственная долгоживущая** ветка







Маленькие и частые PR

достаточно маленькие, чтобы
поместиться в голову ревьювера,
и достаточно сфокусированные, чтобы можно было
понять происходящее **из описания**

получаем **быстрое** и **качественное** ревью
и минимальное отставание от транка



Trunk готов к релизу

новый код и значительные изменения
скрыты за **релизными флагами**

флаги включаются в production **по готовности**
уже **после** попадания кода в транк



Зависимости по исходному коду

вместо публикации пакета в NPM,
используем код в **транке**

облегчаем **разработку** и **использование**
общих решений















Yarn 3

v2+ это Yarn Modern
v1 это Yarn Classic



Использование

устанавливаем **любую** версию глобально,
а использоваться будет **нужная** версия,
сохранённая рядом с кодом проекта

```
$ yarn set version stable
➤ YN0000: Retrieving https://repo.yarnpkg.com/3.2.4/packages/yarnpkg-cli/bin/yarn.js
➤ YN0000: Saving the new release in .yarn/releases/yarn-3.2.4.cjs
➤ YN0000: Done in 1s 4ms
```




Workspaces

<repo>/package.json

```json
{
  "name": "root",
  "workspaces": [
    "@*/*"
  ]
}
```

Project

```
@fc2022
  app
    package.json
  library
    package.json
  package.json
```

Workspace

```
app
  package.json
```




workspace:*

```json
{
  "name": "@fc2022/app",
  "dependencies": {
    "@fc2022/library": "workspace:*"
  }
}
```




yarn workspace

выполняет любую **команду** в контексте **воркспейса** из **любой рабочей директории** в проекте

```
~/fc2022 ❯ yarn workspace @fc2022/app add react
➤ YN0000: ┌ Resolution step
➤ YN0000: └ Completed
➤ YN0000: ┌ Fetch step
➤ YN0000: └ Completed
➤ YN0000: ┌ Link step
➤ YN0000: └ Completed
➤ YN0000: Done in 0s 46ms
```




yarn workspaces foreach

выполняет любую **команду**
в контексте **каждого воркспейса**
с поддержкой **фильтрации,**
последовательно или **параллельно**
с учётом **структуры зависимостей**



yarn workspaces foreach

```
fc2022/@fc2022/app ❯ yarn workspaces foreach \
                        --exclude @fc2022/tools \
                        --recursive \
                        --topological \
                        --parallel \
                        --verbose \
                        run build
➤ YN0000: [@fc2022/api]: Process started
➤ YN0000: [@fc2022/ui]: Process started
➤ YN0000: [@fc2022/api]: Process exited (exit code 0), completed in 0s 40ms
➤ YN0000: [@fc2022/ui]: Process exited (exit code 0), completed in 0s 35ms
➤ YN0000: [@fc2022/app]: Process started
➤ YN0000: [@fc2022/app]: Process exited (exit code 0), completed in 0s 13ms
➤ YN0000: Done in 0s 58ms
```




Plugins

добавляют **новые команды** с доступом к дереву **зависимостей** и **структуре проекта** и **расширяют** поведение Yarn с помощью **хуков**

`@yarnpkg/plugin-typescript`

Features

- Automatically adds `@types/` packages into your dependencies when you add a package that doesn't include its own types



Plugins: commands

```
module.exports = {
  name: `plugin-hello-world`,
  factory: require => {
    const {BaseCommand} = require(`@yarnpkg/cli`);
    class HelloWorldCommand extends BaseCommand {
      static paths = [[`hello`]];
      async execute() {
        this.context.stdout.write(`This is my very own plugin 😎\n`);
      }
    }
    return {
      commands: [
        HelloWorldCommand,
      ],
    };
  }
};
```




Plugins: hooks

```
hooks: {
  afterAllInstalled(project) {
    let descriptorCount = 0;
    for (const descriptor of project.storedDescriptors.values())
      if (!structUtils.isVirtualDescriptor(descriptor))
        descriptorCount += 1;

    let packageCount = 0;
    for (const pkg of project.storedPackages.values())
      if (!structUtils.isVirtualLocator(pkg))
        packageCount += 1;

    console.log(`This project contains ${descriptorCount} different descriptors
                 that resolve to ${packageCount} packages`);
  }
}
```

Plug’n’Play

строгий алгоритм разрешения зависимостей
и установка **без** директории **node_modules**

зависимости сохраняются в **.yarn/cache**
в виде **zip-архивов**

в **.pnp.*** хранится карта зависимостей
с **точными** версиями по **всем** пакетам

при старте **node** инжектится **загрузчик модулей,**
работающий с **.yarn/cache**



Детерминизм

доступ есть **только** к тому,
что указано в **package.json**

нет **фантомных зависимостей**



Фантомные зависимости?

```
"dependencies": {
  "is-even": "^1.0.0"
}
```

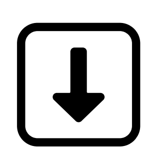
Dependencies (1)

is-odd

```
node_modules
  is-even
  is-odd
```

```
import isOdd from "is-odd";

if (isOdd(n)) { /* */ }
```

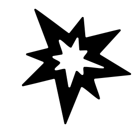


is-even 1.1.0

We are now dependency-free!

```
node_modules
  is-even
```

```
import isOdd from "is-odd";
// Uncaught Error: Cannot find module 'is-odd'
```





Нет node_modules

не нужно **дублировать** вложенную **структуру зависимостей** пакетов в файловой системе

не нужно **обновлять** множество файлов при **изменении** зависимостей

```
node_modules
  .bin
    loose-envify
    rimraf
  @babel
    runtime
      helpers
        esm
          applyDecoratedDescriptor.js
          applyDecs.js
          applyDecs2203.js
          arrayLikeToArray.js
          arrayWithHoles.js
          arrayWithoutHoles.js
          assertThisInitialized.js
          AsyncGenerator.js
```

➡

```
.yarn
  cache
    @babel-runtime-npm-7.19.4-9f106cb4dd-66b7e3c13e.zip
```




Zero-installs

можно **закоммитить кэш** в репозиторий
проект **будет готов** к работе
сразу после **чекаута**
без установки зависимостей

особенно полезно при **частом переключении** между **ветками**

удобнее делать **частые PR**

не нужен кэш зависимостей для **CI**

```
fc2022 ᚴ feature ❯ git checkout main
Switched to branch 'main'
fc2022 ᚴ main ❯ yarn install
➤ YN0000: ┌ Resolution step
➤ YN0000: └ Completed
➤ YN0000: ┌ Fetch step
➤ YN0000: └ Completed
➤ YN0000: ┌ Link step
➤ YN0000: └ Completed
➤ YN0000: Done in 0s 62ms
```




Поломки пакетов

из-за **строгости PnP** пакеты с **фантомными зависимостями** могут ломаться

недостающие зависимости можно добавить с помощью **packageExtensions** на уровне проекта

известные поломки чинятся **автоматически** с помощью встроенного **plugin-compat**

обычно проблем не возникает














VSCode

Один редактор
для всей команды



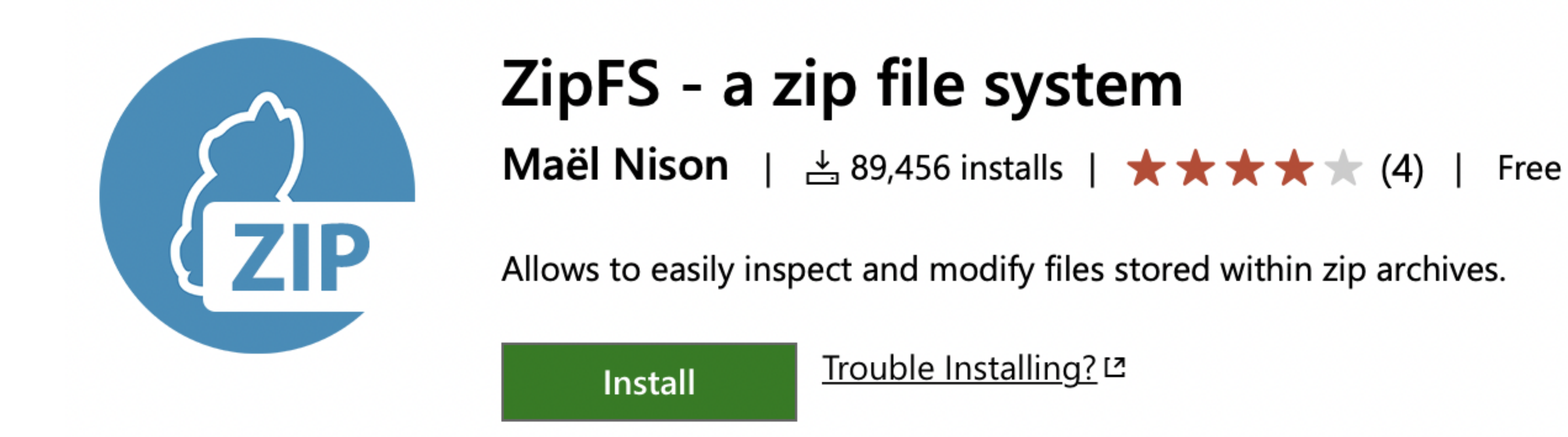
Editor SDK

```
~/fc2022 ❯ yarn dlx @yarnpkg/sdks vscode

➤ YN0000: ┌ Generating SDKs inside .yarn/sdks
➤ YN0000: │ ✓ Eslint
➤ YN0000: │ ✓ Prettier
➤ YN0000: │ ✓ Typescript
➤ YN0000: │ • 3 SDKs were skipped based on your root dependencies
➤ YN0000: └ Completed
➤ YN0000: ┌ Generating settings
➤ YN0000: │ ✓ Vscode (new ✨)
➤ YN0000: └ Completed
```




Работа с архивами





Настройки и использование

коммитим в репозиторий
полезные для **всей** команды **настройки**
и рекомендованные **расширения**

открываем в редакторе **репозиторий целиком:**
ничего не тормозит, у всех **работает одинаково,**
легко работать с **любой частью** кодовой базы

у разработчика готовый к работе **проект** и
настроенное **окружение**















Vite

Бандлер и девсервер

(French word for "quick", pronounced /vit/, like "veet")



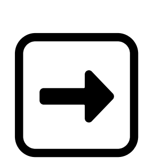
No-bundle в дев-режиме

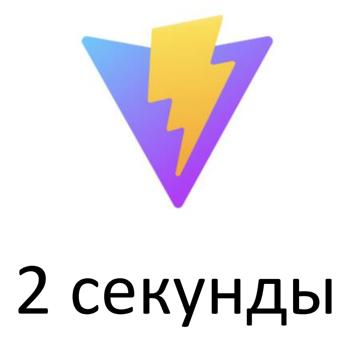
каждый модуль трансформируется
по требованию с помощью **ESBuild**

девсервер стартует **моментально**



2 минуты





2 секунды



Если очень много модулей

когда модулей **действительно много,**
и все они нужны для **отрисовки** страницы,
первая загрузка **после старта** девсервера
может быть относительно долгой

но, скорее всего, **быстрее** бандлера

2k модулей ≈ 5 секунд



Девсервер и бандлер

не только **девсервер,**
но и **бандлер** на основе **Rollup**
с **общими** настройками и поддержкой **плагинов**

совместим с **Rollup-плагинами,**
легко настраивается и расширяется



@vitejs/plugin-react

добавляет поддержку **JSX,**
в том числе новый **automatic** JSX runtime

интегрирует **React Fast Refresh**
во строенный **HMR**



Vite & воркспейсы

имеет **встроенную** поддержку **воркспейсов** и **PnP**

код из **соседних** воркспейсов обрабатывается **так же,** как и код **приложения**

HMR & Fast Refresh работают в том числе в **общих библиотеках**



.env и --mode

```
# .env.dev
VITE_SOME_FEATURE=1

# .env.prod
VITE_SOME_FEATURE=
```

```
export function SomeFeature() {
  return import.meta.env.VITE_SOME_FEATURE
    ? <Implementation />
    : <Fallback />
}
```

```
yarn vite build --mode prod
```




Library Mode

можно собирать **библиотеки** в любом формате

```
export default {
  build: {
    lib: {
      entry: "src/index.ts",
      formats: ["es", "cjs"]
    }
  }
}
```
















Prettier



Prettier

работает в **редакторе кода**

запускается в **CI** для **проверки**
форматирования

снимает все вопросы по оформлению кода

проще использовать **отдельно** от линтера,
чтобы не приходилось **синхронизировать**
конфликтующие настройки



Настройки

общий .prettierignore
в корневой директории

root = true в **.editorconfig,**
чтобы избежать неожиданностей

Editor: Format On Save



Format a file on save. A formatter must be available, the file must not be saved after delay, and the editor must not be shutting down.







TS


TS



TypeScript




FC
Frontend
Conf 2022

Одна версия TS

обеспечивает **консистентность** работы с зависимостями по **исходному коду**

VSCode всё равно будет использовать одну версию



Project references

изолируют части кодовой базы
и позволяют используем разные **опции,**
например, для **node** и **browser**

связывают зависимые части кода,
чтобы **компилятор** и **редактор** могли найти
конфигурацию и код

позволяют **компилировать** общие части
один раз



Project references

- @fc2022
  - app
    - tsconfig.json
  - library
    - tsconfig.json

@fc2022/app/tsconfig.json

```
{
  "references": [
    {"path": "../library"}
  ],
  "compilerOptions": {
    "composite": true,
    "emitDeclarationOnly": true
  }
}
```

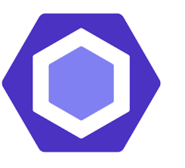
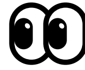
```
yarn tsc --build ./@fc2022/app
```








TS


ESLint




FC
Frontend
Conf 2022

Конфигурация

общий .eslintignore в корне

.eslintrc.js с подключением **плагинов**
и общими **настройками**

не используем сторонние конфиги
формируем **собственный,**
только с **нужными нам** правилами

не включаем стилистические правила
за форматирование отвечает **prettier**



TypeScript ESLint

парсер добавляет поддержку **синтаксиса**

плагин расширяет стандартные правила для работы с **языковыми конструкциями** и предоставляет набор правил с использованием **информации о типах**



TypeScript ESLint

```
module.exports = {
  // подключаем плагин
  plugins: ["@typescript-eslint"],

  // настраиваем парсер
  parser: "@typescript-eslint/parser",
  parserOptions: {
    // указываем пути до всех tsconfig
    project: "./**/tsconfig*.json",

    // включаем экспериментальную опцию
    // для поддержки project references
    EXPERIMENTAL_useSourceOfProjectReferenceRedirect: true,
  },
};
```

Support for Project References #2094



bradzacher opened this issue on May 25, 2020 · 27 comments · Fixed by #2669

feat(typescript-estree): add flag EXPERIMENTAL_useSourceOfProjectReferenceRedirect #2669

Merged **bradzacher** merged 3 commits into `master` from `EXPERIMENTAL_useSourceOfProjectReferenceRedirect` on Oct 15, 2020















Тестирование

Jest & Vitest



Jest



гибко настраивается с помощью раннеров, окружений и трансформеров

конфигурация частично повторяет конфигурацию для сборки

есть **vite-jest** и **esbuild-jest** вместо babel-jest



Vitest



фреймворк для тестирования
на основе **Vite**

переиспользует существующую
конфигурацию и легко **донастраивается**

удобная альтернатива Jest
для **jsdom-** и **node-**тестов

есть примеры e2e с использованием
puppeteer и **playwright**













В итоге



Простое и быстрое переключение

```
fc2022 ᚹ feature-1 ❯ git checkout main
Switched to branch 'main'
Your branch is behind 'origin/main' by 16 commits, and can be fast-forwarded.
  (use "git pull" to update your local branch)
fc2022 ᚹ main [⇣] ❯ git pull
Updating a399130..cc4f3db
Fast-forward
fc2022 ᚹ main ❯ git checkout -b feature-2
Switched to a new branch 'feature-2'
fc2022 ᚹ feature-2 ❯ code .
fc2022 ᚹ feature-2 ❯ yarn workspace @fc2022/app dev

  VITE v3.1.8  ready in 120 ms

  ➜  Local:   http://localhost:5173/
  ➜  Network: use --host to expose
```




Готовое окружение

общие **настройки** и **инструменты**

легко работать над **любой** частью кодовой базы и **переключаться** между ними



стираем технические границы
между проектами



Демо-репозиторий



github.com/swandir/fc2022



оценить доклад



